COMPUTACIÓN EN LA NUBE

Informe Práctica 8

Universidad de La Laguna

INDICE

1. Ejemplos de Spark Streaming. 3

2. Problema a resolver con Spark Streaming. 5

2.1.- Descripción y soluciones. 5

2.2.- Desarrollo y tecnología usada. 6

1. Ejemplos de Spark Streaming.

Para este apartado, se ha seguido el ejemplo que podemos encontrar en el siguiente enlace "https://spark.apache.org/docs/latest/streaming-programming-guide.html". Aquí, nos explica detalladamente cada sección del ejemplo a desarrollar. Este consta, de un programa realizado en Python que nos permite leer cada sentencia escrita por nosotros dentro de la herramienta "Netcat". Para poder realizarlo, se han seguido los siguientes pasos:

- Creamos el StreamingContext con dos subprocesos de ejecución y un intervalo de lote de 1 segundo. Esto lo podremos ver a continuación:

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

# Create a local StreamingContext with two working thread and batch interval of 1 second
sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)
```

- Creamos un "DStream" para representar el envío mediante TCP y separamos los mensajes entrantes por espacios. Este lo podremos ver a continuación:

```
# Create a DStream that will connect to hostname:port, like localhost:9999
lines = ssc.socketTextStream("localhost", 9999)

# Split each line into words
words = lines.flatMap(lambda line: line.split(" "))
```

- Se le asigna (de uno a uno) cada palabra generada a un DStream y, posteriormente imprimimos los recuerdos generados cada segundo. Esto lo podremos ver a continuación:

```
# Split each line into words
words = lines.flatMap(lambda line: line.split(" "))

# Count each word in each batch
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)

# Print the first ten elements of each RDD generated in this DStream to the console
wordCounts.pprint()
```

- Para comenzar el procesamiento después de haber configurado las transformaciones tendremos que añadir las siguientes líneas:

```
ssc.start()             # Start the computation
ssc.awaitTermination()  # Wait for the computation to terminate
```

Tras realizar los pasos anteriores, ejecutamos y comprobamos que todo funciona correctamente. Esto lo podremos ver a continuación:

Código Final

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

# Create a local StreamingContext with two working thread and batch interval of 1 second
sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)

# Create a DStream that will connect to hostname:port, like localhost:9999
lines = ssc.socketTextStream("localhost", 9999)

# Split each line into words
words = lines.flatMap(lambda line: line.split(" "))

# Count each word in each batch
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)

# Print the first ten elements of each RDD generated in this DStream to the console
wordCounts.pprint()

ssc.start()             # Start the computation
ssc.awaitTermination()  # Wait for the computation to terminate
```

NetCat

```
omar@omar-B85M-D3H:~$ nc -lk 9999
Prueba Practica 9 Computacion en la Nube. Usando Spark Streaming
```

Salida programa

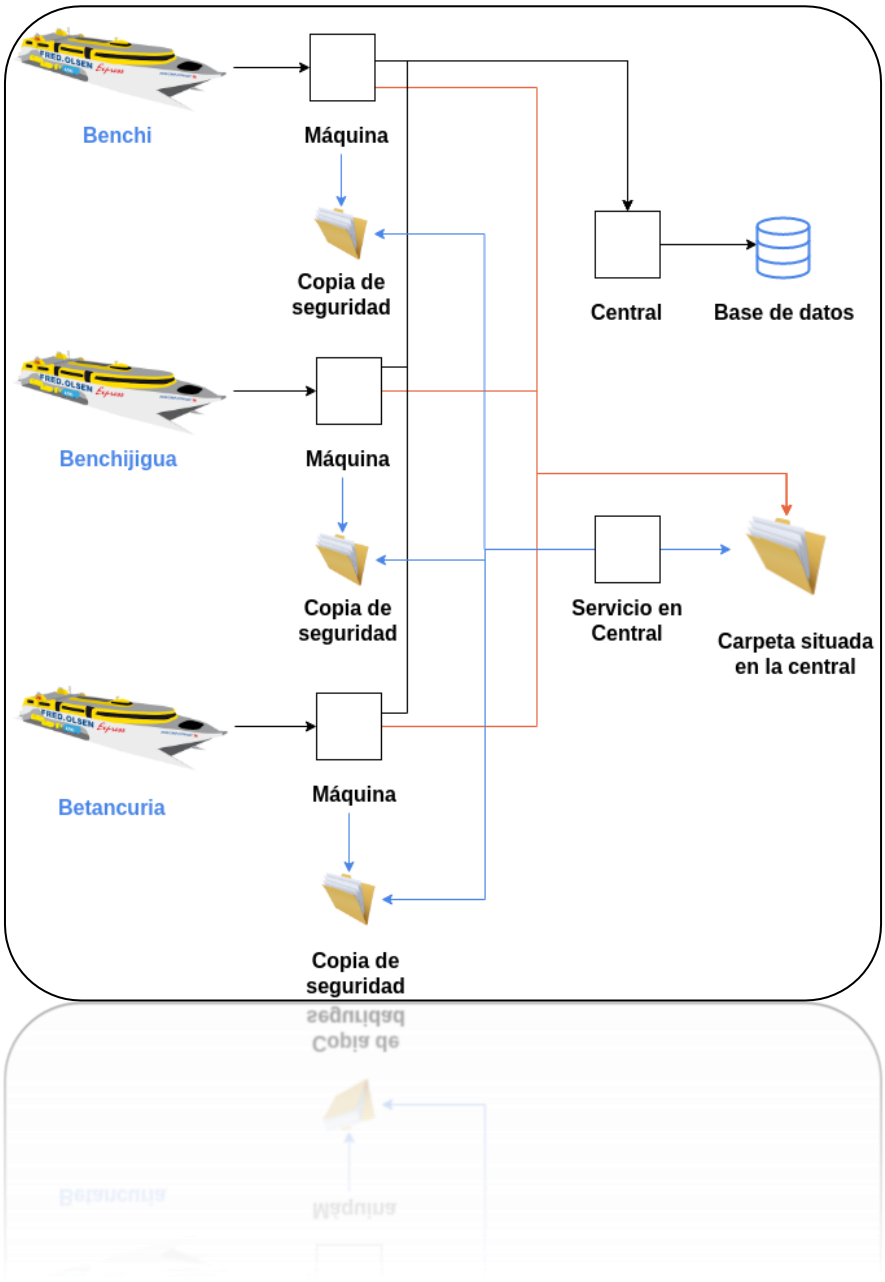
```
-------------------------------------------
Time: 2020-05-16 18:23:20
-------------------------------------------
(u'9', 1)
(u'Spark', 1)
(u'Usando', 1)
(u'en', 1)
(u'Prueba', 1)
(u'la', 1)
(u'Computacion', 1)
(u'Streaming', 1)
(u'Nube.', 1)
(u'Practica', 1)
```

2. Problema a resolver con Spark Streaming.

2.1.- Descripción y soluciones.

Para este ejercicio, se ha pensado en una copia de seguridad para los envíos realizados de una máquina a otra, es decir, una copia de seguridad para la máquina de envío de mensajes.

Este ejercicio, se ha pensado para una posible implantación dentro de la empresa de trabajo (“Fred Olsen”), debido a la importancia del big data dentro de la misma. No obstante, dentro de los buques no se dispone de una máquina muy potente, por lo que se ha pensado en realizar una copia de seguridad de los envíos. Esta idea la podemos ver reflejada a continuación:



Como podemos ver en la imagen anterior, existen dos formas para el envío de la información. Estas son:

- **Forma 1:** sigue los siguientes pasos:
  - La información se va almacenando en una carpeta dentro del buque.
  - Un segundo servicio la almacena dentro de una carpeta compartida en la central de la compañía.
- **Forma 2:** La información se envía directamente a la carpeta compartida de la central.

Como podemos deducir, ambas formas son igualmente válidas. No obstante, se tendrían que probar en la realidad para saber cuál es mejor para este caso particular.

2.2.- Desarrollo y tecnología usada.

Para este ejercicio, se ha usado Python junto con Apache Spark Streaming. Estas tecnologías son los requisitos propuesto por el profesor para esta práctica.

Como vemos en el apartado anterior, es necesario crear un fichero para cada Buque. De igual forma, necesitamos configurarlos. Esto lo podremos ver a continuación:

Benchi

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext


def functionToCreateContext():
    sc = SparkContext("local[2]", "Boats")
    ssc = StreamingContext(sc, 5)
    ssc.checkpoint("./Buques/Benchi")
    return ssc

ssc = StreamingContext.getOrCreate(checkpointPath = "./Buques/Benchi", setupFunc = functionToCreateContext)
lines = ssc.socketTextStream("localhost", 9999)
lines.pprint()

ssc.start()
ssc.awaitTermination()
```

Benchijigua

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext


def functionToCreateContext():
    sc = SparkContext("local[2]", "Boats")
    ssc = StreamingContext(sc, 5)
    ssc.checkpoint("./Buques/Benchijigua")
    return ssc

ssc = StreamingContext.getOrCreate(checkpointPath = "./Buques/Benchijigua", setupFunc = functionToCreateContext)
lines = ssc.socketTextStream("localhost", 9999)
lines.pprint()

ssc.start()
ssc.awaitTermination()
```

Betancuria

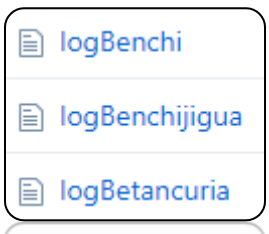
```python
from pyspark import SparkContext
from pyspark.streaming import StreamingContext


def functionToCreateContext():
    sc = SparkContext("local[2]", "Boats")
    ssc = StreamingContext(sc, 5)
    ssc.checkpoint("./Buques/Betancuria")
    return ssc

ssc = StreamingContext.getOrCreate(checkpointPath = "./Buques/Betancuria", setupFunc = functionToCreateContext)
lines = ssc.socketTextStream("localhost", 9999)
lines.pprint()

ssc.start()
ssc.awaitTermination()
```

Para poder simular el envío de información se ha usado la herramienta “Netcat” y, se ha incorporado la información a mano. Un ejemplo de esto lo podremos ver a continuación:

Consola con Netcat

```
omar@omar-B85M-D3H:~$ nc -lk 9999
{ "Temperatura_M1" : "39", “Presion_M1”:”1,5”, "Temperatura_M2" : "37" ,”Presion
”:”2,5”}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
```

Logs Generados



Ejemplo Log Benchi

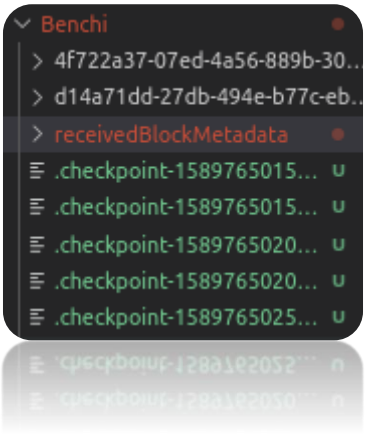
```
-------------------------------------------
Time: 2020-05-18 02:23:55
-------------------------------------------
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
{ "Temperatura_M1" : "39", "Temperatura_M2" : "37"}
```

Como vemos en las imágenes anteriores, se dispone de un log que nos permite guardar los paquetes enviados. No obstante, se dispone de igual forma, un checkpoint y una copia de seguridad avanzada dentro del propio código. Este lo podremos ver en las siguientes imágenes:

Código

```
def functionToCreateContext():
    sc = SparkContext("local[2]", "Boats")
    ssc = StreamingContext(sc, 5)
    ssc.checkpoint("./Buques/Benchi")
    return ssc

ssc = StreamingContext.getOrCreate(checkpointPath = "./Buques/Benchi", setupFunc = functionToCreateContext)
```

Copia de Seguridad



Como podemos ver, este planteamiento cumple con creces la problemática expuesta en el apartado anterior. De igual forma, solo es necesario modificar la línea donde se especifica dónde se van a almacenar los datos (agregando la ruta de dentro del barco o de la central) para realizar las dos opciones planteadas.